UNIVERSIDAD NACIONAL DE INGENIERIA | FACULTAD DE CIENCIAS

Introducción a la Programación

Semana 5:

Funciones en C

Setiembre de 2015

Objetivo

Al finalizar la clase el estudiante debe ser capaz de

- Implementar una función en C
- Emplear funciones como herramienta de la programación estructurada
- Declarar variables locales dentro de una función
- Retornar un valor desde una función al programa

¿Qué es una función?

Es una sección independiente de código en C que realiza una tarea específica y opcionalmente retorna un valor.

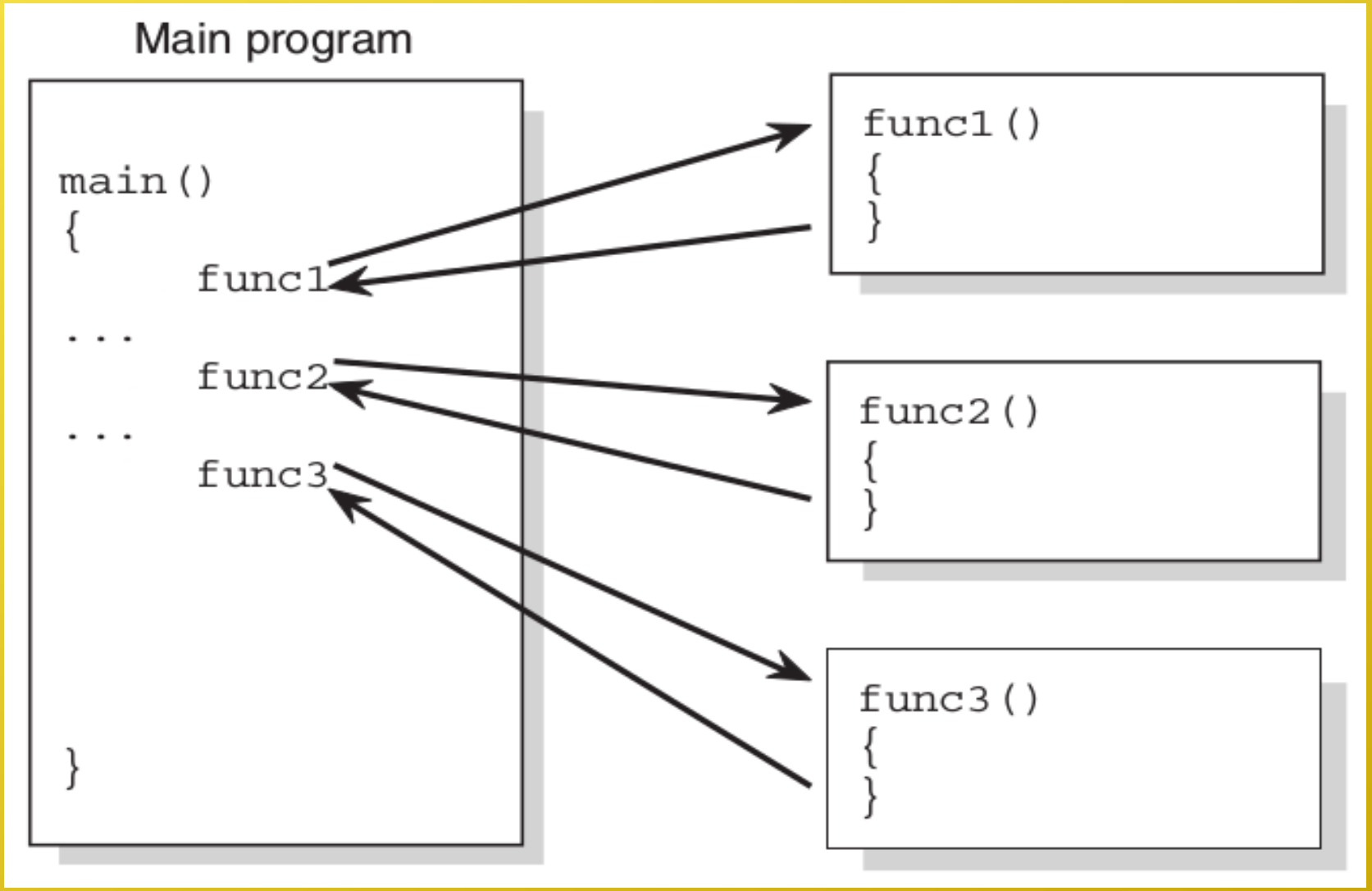
Funciones en C

- Tiene nombre
- Es independiente
- Realiza una tarea específica
- Puede retornar un valor al programa que lo llama

Escribiendo una función

El primer paso para escribir una función es saber qué queremos que haga dicha función.

¿Cómo trabaja una función?



Programa producto.c

```c
/* Este programa pide ingresar dos enteros y muestra
 * el producto de ambos empleando una función */

#include <stdio.h>

int producto (int a, int b); // prototipo de la función producto

void main (void)
{
    int x, y;

    printf("\n Ingrese un valor entero: ");
    scanf("%d", &x);
    printf(" Ingrese otro valor entero: ");
    scanf("%d", &y);
    printf(" El producto de dichos valores es %d\n\n", producto(x,y));
}

// definición de la función producto
int producto (int a, int b)
{
    return a*b; // retornando el valor que resulta de evaluar a*b
}
```

La cabecera de la función

```
2  // Cabecera de la función nombre_funcion
3  tipo_retorno nombre_funcion (tipo_arg_1 nom_1,..., tipo_arg_n nom_n)
4  // Cuerpo de la función nombre_funcion
5  {
6      sentencia;
7  }
```

Está constituida por tres componentes:

- El **tipo de retorno de la función**: tipo de dato de C
- El **nombre de la función**: un identificador
- La **lista de parámetros**: tipo de dato de los argumentos pasados a la función

¿Parámetro y argumento es lo mismo?

Un **parámetro** es una entrada en la cabecera de una función y sirve como un reservador de espacio para un argumento; mientras que un **argumento** es un dato que es pasado a una función.

El cuerpo de la función

```
2  int producto (int a)
3  // Cuerpo de la función producto
4  {
5      int x = 2, y; // variables locales
6      y = x*a;
7      return y; // retornando un valor
8  }
```

El cuerpo de la función está entre llaves y es lo que sigue inmediatamente después de la cabecera de la función.

El cuerpo de la función

```
2  int producto (int a)
3  // Cuerpo de la función producto
4  {
5      int x = 2, y; // variables locales
6      y = x*a;
7      return y; // retornando un valor
8  }
```

Es en el cuerpo de la función que se lleva a acabo el trabajo: cuando una función es llamada, la ejecución comienza al inicio del cuerpo y termina cuando una **sentencia return** es hallada o cuando se alcanza }.

El cuerpo de la función

```
2  int producto (int a)
3  // Cuerpo de la función producto
4  {
5      int x = 2, y; // variables locales
6      y = x*a;
7      return y; // retornando un valor
8  }
```

También se pueden declarar variables dentro del cuerpo de una función, estas son llamadas **variables locales**.

El prototipo de la función

```
2  // prototipo de la función producto
3  int producto (int a, int b);
4
5  // Cabecera de la función producto
6  int producto (int a, int b)
7  {
8      // cuerpo de la función
9      sentencia;
10 }
```

Un programa debe incluir un prototipo para cada función que utiliza. El trabajo del prototipo de una función es el de proveer al compilador información acerca de esta función: valor de retorno, nombre y parámetros.

El prototipo de la función

```
 2  // prototipo de la función producto
 3  int producto (int x, int y);
 4
 5  // Cabecera de la función producto
 6  int producto (int a, int b)
 7  {
 8      // cuerpo de la función
 9      sentencia;
10  }
```

Así, el compilador podrá hacer su trabajo de revisar cada vez que la función es llamada en el programa. Los prototipos deben ser ubicados antes del inicio de la primera función.

Funciones y la programación estructurada

Empleando funciones en nuestro programa podemos practicar la **programación estructurada**, en el que tareas individuales del programa son llevadas a cabo por secciones independientes de código.

Ventajas de la programación estructurada

- Es más fácil escribir un programa estructurado porque los problemas de programación más complejos pueden ser divididos en problemas más sencillos.

- Es más sencillo depurar un programa estructurado porque es más fácil aislar el problema a una sección específica de código.

Planeando un programa estructurado